



HAL
open science

Contribution of the Web of Things and of the Opportunistic Computing to the Smart Agriculture: A Practical Experiment

Lionel Touseau, Nicolas Le Sommer

► **To cite this version:**

Lionel Touseau, Nicolas Le Sommer. Contribution of the Web of Things and of the Opportunistic Computing to the Smart Agriculture: A Practical Experiment. *Future internet*, 2019, 11 (2), pp.33. 10.3390/fi11020033 . hal-02307171

HAL Id: hal-02307171

<https://hal.science/hal-02307171>

Submitted on 7 Oct 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Article

Contribution of the Web of Things and of the Opportunistic Computing to the Smart Agriculture: A Practical Experiment

Lionel Touseau and Nicolas Le Sommer

{Lionel.Touseau, Nicolas.Le-Sommer}@univ-ubs.fr
IRISA, Université Bretagne Sud

Version January 28, 2019 submitted to Future Internet

1 **Abstract:** With the emergence of the Internet of Things, environmental sensing has been gaining interest,
2 promising to improve agricultural practices by facilitating decision-making based on gathered environmental
3 data (i.e., weather forecasting, crop monitoring, soil moisture sensing). Environmental sensing, and by extension
4 what is referred to as precision or smart agriculture, pose new challenges, especially regarding the collection of
5 environmental data in presence of connectivity disruptions, their gathering, and their exploitation by end-users
6 or by systems that must perform actions according to the values of those collected data. In this paper, we
7 present a middleware platform for the Internet of Things that implements disruption tolerant opportunistic
8 networking and computing techniques, and that makes it possible to expose and to manage physical objects
9 through Web-based protocols, standards and technologies, thus providing interoperability between objects and
10 creating a Web of Things (WoT). This WoT-based opportunistic computing approach is backed up by a practical
11 experiment whose outcomes are analyzed in this article.

12 **Keywords:** Web of Things, Intermittent Connectivity, Opportunistic Computing, Opportunistic Sensing,
13 Precision Agriculture, Smart Agriculture

14 1. Introduction

15 With the recent progress achieved in the automation of farm machinery and in the monitoring of the physical
16 environment, the agriculture moves towards a more environmental-friendly, smart and precise agriculture,
17 addressing challenges ranging from manpower shortage, adaptation to climate changes, optimization of
18 environment resource usage, reduction of the usage of chemical crop protection products, etc. The Internet of
19 Things (IoT) – which aims at connecting to the Internet physical objects that can sense, communicate, compute
20 and sometimes actuate – can indubitably contribute to the development of the smart agriculture and farming. A
21 few number of research works have recently experimented the usage of IoT platforms in the agriculture and
22 farming context [1–3]. These experiments have highlighted several issues that must be addressed to efficiently
23 and automatically collect data in such challenging contexts, especially regarding the network connectivity, the
24 network architecture and the interoperability between physical objects. Applications and services dedicated to
25 smart agriculture and farming can tolerate, in some cases, delays and disruptions in data exchanges. Indeed, some
26 physical objects do not need to be connected to the Internet permanently, because they measure environmental
27 properties that do not evolve quickly (e.g. soil moisture, plant growth), and because they are in sleep mode most
28 of the time for energy saving purposes. An alternative or a complement to long range, low power and low rate
29 wireless standards such as LoRa/LoRaWAN and SigFox, could be to interconnect such devices to gateways
30 opportunistically using data mules and short radio range wireless interfaces such as IEEE 802.15.4 and IEEE
31 802.11 — that allow to transfer, thanks to a higher throughput, a significant amount of data in a short time
32 from sensors (e.g. camera) or to agribots (field maps, tasks to achieve, etc.). Such an alternative, that relies on
33 opportunistic communication and computing techniques, is known as the opportunistic sensing paradigm [4].

Besides these problems, the lack of semantic description and logical representation of physical objects does not allow to easily consolidate collected data with data from external Web services (e.g. weather forecast, plant growth models), to assemble objects to build sophisticated systems, and to create high level services and applications that should help farmers to take decisions, or even to perform tasks automatically. The Web of Things (WoT) [5] is an attractive solution to address these issues. Indeed the WoT extends the IoT, and aims at doing for physical objects what the Web did for information resources, namely an identification of resources using Uniform Resource Identifiers (URIs), a semantic description of resources, an access of resources through standard protocols, and the inclusion in resources of links to other resources in order to enable a scalable discovery of highly distributed resources. Following the trend of Web 2.0 participatory services, in particular Web Mashups, it thus is possible to create applications mixing physical devices with virtual services on the Web — this type of applications is often referred to as physical Mashup [6,7].

In this paper, we present a WoT oriented platform, called ASAWoO¹ [8], that implements opportunistic and disruption-tolerant networking and computing techniques. This platform makes it possible to define and instantiate logical extensions, called Avatars, of physical objects. Physical objects can be configured and controlled through their avatar. Avatars provide a semantic description of physical objects and of their functionalities. Functionalities are implemented as REST services, and can be deployed dynamically by the avatars themselves according to the description of the physical objects and of the execution context of these ones. Like functionalities, execution contexts are described semantically, thus allowing avatars to reason on these descriptions and to perform adaptations dynamically. REST services can be invoked using standard protocols such as HTTP or CoAP, either using Internet-legacy transport protocols or disruption-tolerant and opportunistic protocols. This platform can be deployed on physical objects themselves provided that they enough processing and memory capacities, on gateways to which physical objects are connected or on cloud infrastructure. Objects can be fixed or mobile. For instance, data generated by the fixed sensors can be collected by data mules (e.g., tractors) and transmitted to their logical representations deployed in a cloud infrastructure. These features make this platform a relevant support for the development and the deployment of cyber-physical systems and applications dedicated to the environmental sensing and to the smart agriculture. In this paper, we also report experiments we achieved with this platform in an experimental agricultural farm.

The remainder of this paper is structured as follows. Section 2 presents research works dealing with opportunistic environmental sensing, and details their limitations. Section 3 presents the ASAWoO middleware platform, the concept of avatars, and discuss of the advantages such a platform could bring in an opportunistic sensing and actuating context like the smart agriculture and farming. Section 4 describes the experiments we performed with the ASAWoO platform, and presents some results we obtained. Section 5 concludes this paper by summarizing our contribution.

2. Related work

Over the last years, a large majority of research works dealing with agricultural crop monitoring have focused on the development of small-scale testbeds and specialized applications built on wireless networks formed by fixed sensors (WSN) [3,9,10]. [11] is an example of such works that rely on an infrastructure, where all the devices – a meteorological station, a sprinkler system and humidity sensors deployed a the field, a base station installed near the field and connected to the network infrastructure, which plays the role of gateway between the network of sensors and the infrastructure – are in radio range. Some works consider the issues of distances and of intermittent connections – mainly induced by the duty cycles of sensors for energy saving purposes – between the devices, and solves them by adding redundant relays and gateways so as to build a connected network. A few number of works dealing with precision and sustainable agriculture propose to exploit the mobility of devices carried by people or embedded in agricultural vehicles (e.g. farm tractors, agribots) in order to collect data generated by sensors or to deliver data to actuators (e.g. sprinkler system, agribots). Such a

¹ <https://asawoo.gitlab.io/>

79 collection process, that leverages the contact opportunities between devices, is referred as being opportunistic
80 sensing.

81 Opportunistic sensing and communication techniques, which have an indubitable interest in the agricultural
82 domain, have also been considered in a few number of systems that propose to exploit mobile devices (carried by
83 people or embedded in vehicles) to collectively measure social or environmental data in large-scale pervasive
84 environments. These systems rely on opportunistic and people-centric sensing techniques. They differ from
85 first-generation sensor networks, not only in their goal to support concurrent people-centric and opportunistic
86 sensing applications in urban environments, but also in their hardware and software heterogeneity, high volatility,
87 and very large scale. CarTel [12], MetroSense [13], MobiScopes [14], SenseWeb [15], Urbanet [16] and Urban
88 Atmospheres [17] are examples of such systems. To report information from a given region using these systems,
89 applications run sensing tasks on the mobile devices that are located in this region and that have chosen to
90 participate to the sensing process. Mobile devices report sensor data through opportunistic network connections,
91 such as those they establish with third-party access points existing in their radio range. For instance in CarTel,
92 a network node is a mobile device coupled to a set of sensors. Each node gathers and processes sensor data
93 locally before delivering them to a central portal on the Internet, where the data are stored in a database for
94 further analysis and visualization. Mobile devices opportunistically communicate, using their wireless interface
95 (e.g. Wi-Fi, Bluetooth), with other CarTel mobile devices and with Internet access points found in urban areas.
96 The portal and the mobile nodes use a delay-tolerant network stack, called CafNet, to communicate. CarTel
97 applications run on the portal, and submit SQL queries to a delay-tolerant continuous query processor called
98 ICEDB. This query processor maintains a list of queries submitted by the applications, and pushes them to remote
99 nodes using CafNet. It receives results from remote nodes, and puts them into a relational database. MetroSense
100 adopts a similar approach, and envisions a future Internet in which a large part of the data traffic is generated by
101 sensors carried by people during their daily life. BikeNet [18] and CenceMe [19] are examples of applications
102 based on the MetroSense system. In BikeNet, bicycles equipped with multiple sensors communicated with
103 neighboring bikes as well as the network infrastructure, and deliver sensing data to a Web portal that promotes
104 social networking among cyclists. Similarly in CenceMe, mobile phones collect data about a user's activity and
105 share it with buddies via a social network. Several other sensing applications have been proposed by project
106 Urban Atmospheres [17] and the MIT Senseable City Lab [20]. Project SenseWeb proposes a Web-based
107 platform and tools to publish and query sensor data across Internet. Sensors can measure various physical
108 variables, can be static or mobile and carried by humans, or embedded in vehicles or on robots. To hide the
109 heterogeneity and the intermittent connectivity of sensors, sensors are connected to a gateway that provides a
110 uniform interface to communicate with them. Urbanet also focuses on sensing in spontaneous urban networks
111 composed of heterogeneous and potentially mobile sensors. But unlike the above-mentioned works, it does not
112 rely on central collection points across the Internet that perform data and task management, and act as mediators
113 between users and the network. Urbanet proposes three middleware solutions to develop distributed sensing
114 applications that don't require servers or Internet connectivity; sensing applications running on mobile devices.
115 These solutions respectively present the network 1) as a distributed sensor database that can be queried via a
116 SQL-like language, 2) as a single virtual name space that applications use to access individual resources offered
117 by nodes, and 3) as a client-service model where services migrate to different network nodes to maintain a
118 semantically correct and continuous interaction with clients.

119 The above-mentioned works have drawbacks regarding the opportunistic protocol they implement, as well
120 as Web-based approach they propose. Indeed like CafNet [21], opportunistic protocols used in these works
121 often rely on a simple epidemic (nodes exchange with the other nodes they encounter all the messages they
122 have). Yet, such an epidemic approach is known to not be the more efficient one, especially when there is a
123 lot of data to exchange; more sophisticated and efficient opportunistic protocols have indeed been proposed
124 to address the issues of the epidemic routing protocol [22]. Moreover, these works propose a Web portal that
125 renders the data that have been collected from the sensors, and do not propose logical extensions of physical
126 objects in the Web that are able to provide a semantic description of these objects and of their functionalities, and
127 that offer means to configure and to control them through Web standards and protocols. Yet, as mentioned in
128 the introduction of this paper, such an approach offer significant advantages; the physical Mashup being one

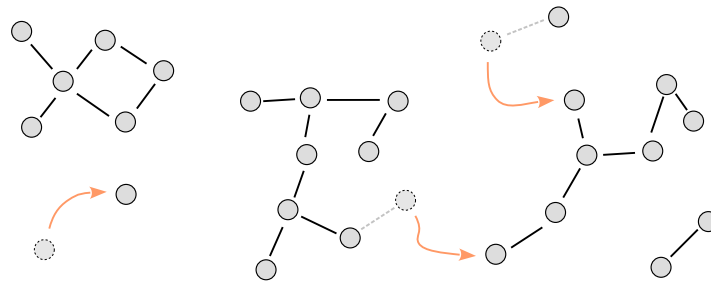


Figure 1. Opportunistic networking in a disconnected mobile network.

129 of them. WoT solutions have been investigated in the agricultural context [23,24] (and also in the smart home
 130 context [25,26]) in order to improve the processing of dataflows produced by sensors and to expose API of
 131 physical objects in the Web. Agri-IoT [23] is a data analytic platform composed of multiple layers, allowing
 132 to communicate with devices (lower layer), to analyze data (intermediate layers) and to present information
 133 to end-users either using dashboards or mobile applications (higher layer). In this platform, sensors, and the
 134 data they produce, are semantically described using agricultural ontologies. But again, these solutions do not
 135 implement disruption-tolerant or opportunistic networking techniques to cope with the connectivity disruptions
 136 that can occur in smart agriculture and farming application domains, and more generally in scenarios involving
 137 mobile devices communicating with short range wireless interfaces.

138 In the next section, we present a WoT dedicated platform called ASAWoO. This platform implements
 139 opportunistic networking and computing mechanisms, and addresses the drawbacks of above-presented works.

140 **3. Leveraging the Web of Things and the opportunistic computing to build systems for the smart** 141 **agriculture**

142 The smart agriculture mainly consists in growing seedlings and plants while preserving natural resources
 143 (e.g. water) and limiting crop protection products. To do so, both environmental properties (soil moisture,
 144 ambient temperature, sunshine, ...) and crop growth are monitored using fixed sensors or sensors embedded on
 145 field machines, drones or agribots. Collected data are compared to agricultural data and models to make decisions
 146 about the treatments to be performed and the means to achieve them (traditional agricultural machines, agribots,
 147 drones). We argue that models, protocols, techniques and approaches developed both in the WoT and in the
 148 opportunistic computing can help develop and deploy cyber-physical systems and applications dedicated to the
 149 smart agriculture. In the remainder of this section, we develop and justify these words by explaining successively
 150 how the opportunistic computing and the WoT can contribute to the smart agriculture. Before detailing these
 151 contributions, we remind the key features of the opportunistic computing and of the WoT.

152 *3.1. Contribution of the opportunistic computing*

153 This subsection first explains what opportunistic computing is and to what extent it could benefit smart
 154 agriculture. Then it thoroughly describes how a support for opportunistic computing was implemented in a Web
 155 of Things platform.

156 *3.1.1. Definitions of opportunistic computing and sensing, and their suitability for smart agriculture*

157 Opportunistic networking is a research field that can be considered as being a derivative of the Delay-Tolerant
 158 Networking (DTN) research domain. While DTN [27] characterizes networks with long-delay communications
 159 and/or with an unreliable connectivity, opportunistic networks denote mobile networks that exploit transient
 160 – and sometimes non-predictable – radio contacts between mobile devices to exchange data [28]. Opportunistic
 161 networking therefore mostly targets disconnected mobile ad hoc networks (D-MANETs) as illustrated on Figure 1.
 162 Yet the differences between these two approaches, which both rely on the “store-carry-and-forward” principle,
 163 are marginal, and both terms are indistinctly used in this paper. Numerous opportunistic forwarding protocols
 164 have been proposed over the past years [22], investigating forwarding strategies based notably on epidemic
 165 approaches or on social criteria (computed from mobility patterns, history of contacts or location information).

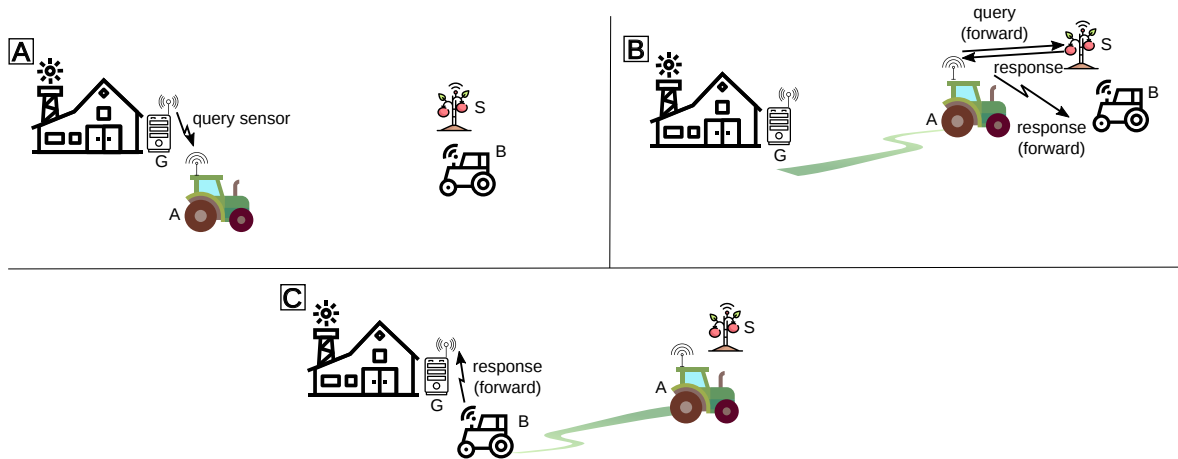


Figure 2. Opportunistic Sensing in a crop monitoring scenario.

166 The opportunistic computing [29] is an extension of the opportunistic networking, where resources offered by
 167 devices are, in general, abstracted and made available through services, thus allowing devices to access resources
 168 provided by the devices they have encountered directly or *via* intermediate devices. Opportunistic computing
 169 and networking do not make assumptions about the network topology and support frequent and unpredictable
 170 changes in this one, as well as its fragmentation.

171 Such features make opportunistic computing and networking interesting for the development of systems
 172 dedicated to the smart agriculture. Unlike urban areas that may offer solid network infrastructures favorable
 173 to environmental sensing, farms are usually located in rural areas where the cellular network coverage is often
 174 lackluster and therefore does not provide a reliable network infrastructure. Furthermore devices involved in smart
 175 agriculture (e.g. sensors, agribots, drones) are by nature likely to fall in the D-MANET category, because devices
 176 may run out of battery or enter a sleep mode for energy-saving purposes, and because mobile devices enter
 177 and exit the communication radius of other nodes as they roam. Computing systems relying on opportunistic
 178 communications though make it possible to collect data generated by sparsely distributed sensors, even if these
 179 ones are not up at the same time and/or if it does not exist an end-to-end path between some sensors and the
 180 gathering point. For this purpose, mobile devices (e.g. tractors, agribots, drones, smartphones carried by farmers)
 181 can in addition be used as “data mules” to collect data from sensors directly or to transfer data between the
 182 different parts of the network that are isolated from one another. Opportunistic computing furthermore allows
 183 farmers to deploy sensors or actuators in their fields and/or in their farms, even on their animals [30], without
 184 having to worry about the location, the radio range, the duty cycle and the sleep phases of these devices.

185 For the sake of illustration, let us consider a crop irrigation management application that can process soil
 186 moisture data produced by sensors deployed in agricultural fields, and that can trigger the watering system if the
 187 soil is too dry and if no rainfall is expected soon. As shown in Figure 2 two farming vehicles A and B, that act as
 188 data mules in this smart agriculture scenario, can participate to the collect of data and to their transmission to the
 189 gateway, where they will be analyzed by a WoT application in order to decide if the watering system must be
 190 triggered or not. In Figure 2.A gateway G emits a service invocation message to query sensor S. This message is
 191 stored and carried by the tractor A. In Figure 2.B the tractor enters in the radio range of sensor S, and delivers the
 192 message to S. Finally, in Figure 2.C the service response sent by sensor S is forwarded back to the gateway G by
 193 the tractor B which returns to its shelter located in the gateway vicinity. The experiment further presented in this
 194 paper implements a similar scenario.

195 3.1.2. Opportunistic computing implementation in ASAWoO Web of Things platform

196 In project ASAWoO, we have designed and developed a disruption-tolerant RESTful support for the
 197 WoT [31]. In this opportunistic computing implementation, resources offered by physical objects are identified by
 198 URIs and accessed through stateless services. Service requests and responses are forwarded using the “store, carry
 199 and forward” principle. This communication support provides a complete service invocation model, allowing to

200 perform unicast, anycast, multicast and broadcast service invocations either using HTTP or CoAP, which makes
 201 it particularly suited for the WoT. It is compliant with the six constraints –the sixth one is optional– defined by
 202 the REST architecture style for performance, scalability and simplicity purposes. Indeed, it does not maintain any
 203 session state, and implements a loosely-coupled client/server architecture where resources offered by physical
 204 objects are accessible through well defined service interfaces. By implementing the “store, carry and forward”
 205 principle, it makes it possible to store service responses in the cache of clients and of intermediate nodes, but also
 206 the service requests that have been sent in the network. Intermediate hosts equipped with this communication
 207 system can also act as proxies, and can respond on behalf of a server if they have the response in their local cache,
 208 when this one is still valid. Such an approach improves the performance and the scalability of the system, because
 209 it naturally performs load balancing and data caching on intermediate hosts, and thus fulfills both the “system
 210 layering” requirement and the “response cacheability” requirement promoted by the RESTful architecture style.
 211 Finally, as WoT applications can be partially or fully developed in Javascript, a part of the application can be
 212 executed on the client side, thus allowing to be compliant with REST optional “code-on-demand” constraint, and
 213 to reduce the computation load on the physical objects.

214 Like most of the disruption-tolerant communication systems, our system implements a multiple copy
 215 forwarding strategy. It can thus take advantage of these message transmission models to increase the message
 216 delivery probability and to reduce the response time in a WoT context. For instance, several sensors can
 217 simultaneously be invoked using a multicast transmission model without naming them explicitly. All the
 218 responses returned by these sensors will be transmitted to the client.

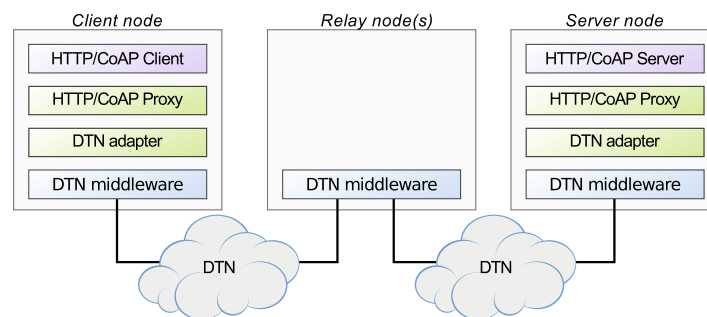


Figure 3. Overview of the implementation of the RESTful opportunistic computing middleware support.

219 The system we have developed allows to invoke services using the HTTP and CoAP application-level
 220 protocols. The application-level messages (i.e. HTTP and CoAP messages) can be encapsulated in UDP
 221 datagrams, in TCP segments or in messages of a given disruption-tolerant communication system in order
 222 to be transmitted to their destination. Different wireless technologies (e.g. Bluetooth, Wi-Fi, zigbee) can be
 223 employed to communicate with physical objects. As shown in Figure 3, this system is implemented by two
 224 main elements, namely an HTTP/CoAP proxy and a DTN adapter. The HTTP/CoAP proxy makes it possible
 225 for standard HTTP and CoAP clients and servers to send and receive service requests and responses using
 226 disruption-tolerant communication techniques. Thanks to this proxy, programmers can develop HTTP and CoAP
 227 WoT applications using regular HTTP and CoAP, libraries. Moreover, standard HTTP and CoAP servers do not
 228 need to be modified. This proxy can also invoke remote REST services using Internet-legacy routing protocols
 229 (i.e. TCP/IP). The HTTP/CoAP proxy is a common element shared between all the implementations. As for
 230 the DTN adapter, it binds the proxy and the disruption-tolerant communication system in charge of forwarding
 231 messages in the network. Hence, the DTN adapter depends on the underlying communication system and is
 232 specifically developed for each different system. Two adapters have been developed: one relying on the Bundle
 233 Protocol (BP) [32], which is the standard message-based protocol over the DTN architecture [33], and another
 234 one relying on the C3PO opportunistic communication platform C3PO [34]. This platform provides interesting
 235 features to implement the anycast, multicast and broadcast communication models efficiently. For instance,
 236 service responses are used to stop the propagation of service requests in the network, and messages can be
 237 restricted to a given geographical area. In the experiment presented in the next section, we used the C3PO
 238 adapter. The original and efficient features implemented in this RESTful opportunistic computing middleware

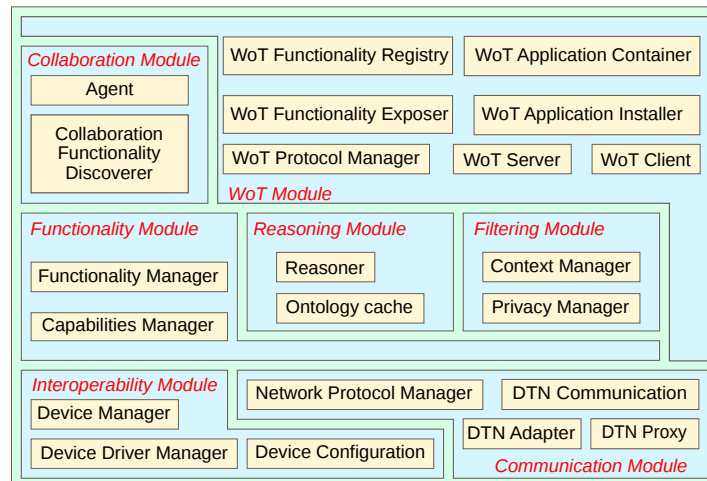


Figure 4. Architecture of an avatar.

239 support allows platform ASAWoO to not have the same drawbacks as those presented in section 2, and make it
 240 an interesting platform adapted to the opportunistic sensing and to the smart agriculture and farming.

241 3.2. Contribution of the Web of Things

242 The Web of Things allows to control and to communicate with physical objects through the logical software
 243 entities that represent them in the Web, and via Web standards and protocols. These logical software entities
 244 allow 1) to describe the objects, 2) to syntactically and semantically describe functionalities (i.e. the Web
 245 services) providing access to hardware capabilities of the physical objects, 3) to establish communications with
 246 the physical objects, by eventually adapting the Web protocols and standards to the proprietary protocols and data
 247 formats used by the objects, 4) to provide environment to deliver and to run WoT applications. Thanks to these
 248 logical software entities, physical objects can be combined together using Web mashup techniques so as to create
 249 complex cyber-physical systems that can be, for instance, designed for the smart agriculture. For instance, a WoT
 250 application built on the logical extensions of a crop irrigation system, of sensors deployed in fields to measure soil
 251 moisture, and of a weather forecast Web service available on the Internet, could easily be developed using mashup
 252 techniques so as to process data produced by sensors, and to trigger the watering system if the soil is too dry
 253 and if no rainfall is expected soon. As mentioned before, the data generated by the sensors can be collected by
 254 data mules (e.g., tractors) and transmitted to their logical representations deployed in a cloud infrastructure. The
 255 application can query these logical representations to obtain the environmental measures, can invoke a forecast
 256 Web service available on the Internet, and can decide on the basis of these pieces of information to trigger the
 257 watering or not. If so, this application will act on the watering system through its logical representation.

258 In the remainder of this section, we present the concept of avatar [35] and the middleware platform we have
 259 proposed in project ASAWoO, and how they can be used in a smart agriculture scenario similar to that described
 260 in the previous lines. We also give a comparison of the concept of avatars with the concept of servient proposed
 261 by the W3C [36].

262 3.2.1. Overview of the Avatar concept

263 The purpose of an avatar is to provide device vendors, software developers and end-users with a
 264 comprehensible abstraction that makes physical objects accessible on the Web and that extends their status and
 265 capabilities (processing, acting, sensing, etc.) into homogeneous and user-understandable functionalities. An
 266 avatar can be viewed as a "Web-based cyber-physical object" that defines and embodies high-level behaviors for
 267 physical devices.

268 An avatar is composed of 7 modules (see Figure 4). The interoperability module makes it possible to
 269 configure and to manage the physical object linked to the avatar. It acts as an adaptation layer that hides
 270 objects' own protocols. The communication module provides avatar-to-avatar and avatar-to-Web-browser

271 communications. Different kinds of applicative protocols (e.g. HTTP, CoAP) and transport protocols (UDP,
272 TCP, the disruption-tolerant protocol provided by the opportunistic communication support presented previously
273 in this section) are supported to communicate with the avatars. Moreover, different wireless technologies
274 are currently supported to achieve communications with physical objects, such as Bluetooth, Wi-Fi, Wi-Fi
275 Direct and XBee. The functionality module manages the capabilities and the functionalities of an avatar. Like
276 functionalities, capabilities are pieces of codes that make it possible to exploit the resources offered by the
277 physical objects. But unlike functionalities – which are device independent –, capabilities depend on the physical
278 objects. They can be viewed as implementations of elementary functionalities. Thanks to this approach, an agribot
279 can for instance expose a higher-level of abstraction for the navigation functionality that exploits lower-level
280 functionalities, such as GPS positioning and steering capabilities. Both capabilities and functionalities are
281 described semantically. Reasoning about semantic descriptions of the capabilities and of the functionalities help
282 inferring complex functionalities involving sub-functionalities, thus allowing to define high-level representations
283 that better match the users' needs than the low-level capabilities. Based on the inferences made by the semantic
284 reasoner, the functionality manager can incrementally deploy new functionalities. These functionalities are
285 filtered by the context manager according to the information it processes continuously, thus allowing to select the
286 implementations of the functionalities that are the most suited to the running conditions of the physical object, as
287 well as to perform dynamic adaptations. Various contextual properties can be taken into account, such as the
288 location of the object, its processing and memory capacities and its power budget. Functionalities are exposed as
289 REST services by the WoT module, and therefore can be invoked remotely by WoT applications (WoTApps) or
290 by other avatars for collaboration purposes. The WoT application container deploys and manages the life-cycle of
291 WoTApps. WoTApps can fully run server-side (i.e. on the avatar), be distributed and executed both on the avatar
292 and in a Web browser, or be exclusively run in a Web browser. Finally, avatars implement a collaboration module
293 relying on an agent-based approach. This module aims at endowing an autonomous behavior to the avatar and
294 physical object, so that they can discover the functionalities they can partially achieve and that require the help of
295 other avatars to be performed. After discovering the missing functionality in another avatar, they can enroll this
296 avatar to provide the functionality.

297 The main differences between the proposition of the W3C for the WoT, called Servient [36], and the
298 avatar-based model reside in the ability of avatars to tolerate the connectivity disruptions, to dynamically
299 adapt their behavior and the functionalities they provide to their running context, to automatically deploy new
300 functionalities inferred on the basis of the semantic description of more elementary ones, and to collaborate
301 with other avatars. Indeed like avatars, servients implement a legacy communication module that makes it
302 possible to communicate with physical objects using proprietary protocols, a protocol binding module that
303 allows to use several types of applicative protocols (MQTT, HTTP, CoAP) to communicate with the servients,
304 a model of resources and of description of objects, and a runtime environment to run WoTApps. These
305 differences between avatars and servients make avatars more suited to develop cyber-physical systems that
306 integrate autonomous mobile objects, and therefore more adapted to systems dedicated to the smart agriculture
307 and farming. Furthermore, the reference implementation of the W3C standard is not as mature as that proposed
308 by the ASAWoO project. Unlike solutions proposed in works [23–26], both Servients and Avatars provide users
309 and developers with a comprehensive virtual representation of physical objects to access and control them. This
310 unified resource representation helps coping with device heterogeneity, which is not well supported in Agri-IoT
311 according to its authors.

312 3.2.2. Avatar runtime environment

313 In order to configure themselves automatically, avatars rely on a set of semantic and code repositories
314 as represented on Figure 5. These repositories contain device setups, semantic descriptions of capabilities
315 and functionalities of avatars and semantic descriptions of execution contexts. The code repositories provide
316 independent syntactic descriptions of REST interfaces of functionalities, and specific implementations of
317 capabilities, functionalities and WoTApps. Most of capabilities are device dependent, which is not the case of
318 the functionalities. The implementations of functionalities can still differ for non functional purposes such as
319 quality of service requirements. Avatars download pieces of code from the code repositories and deploy them

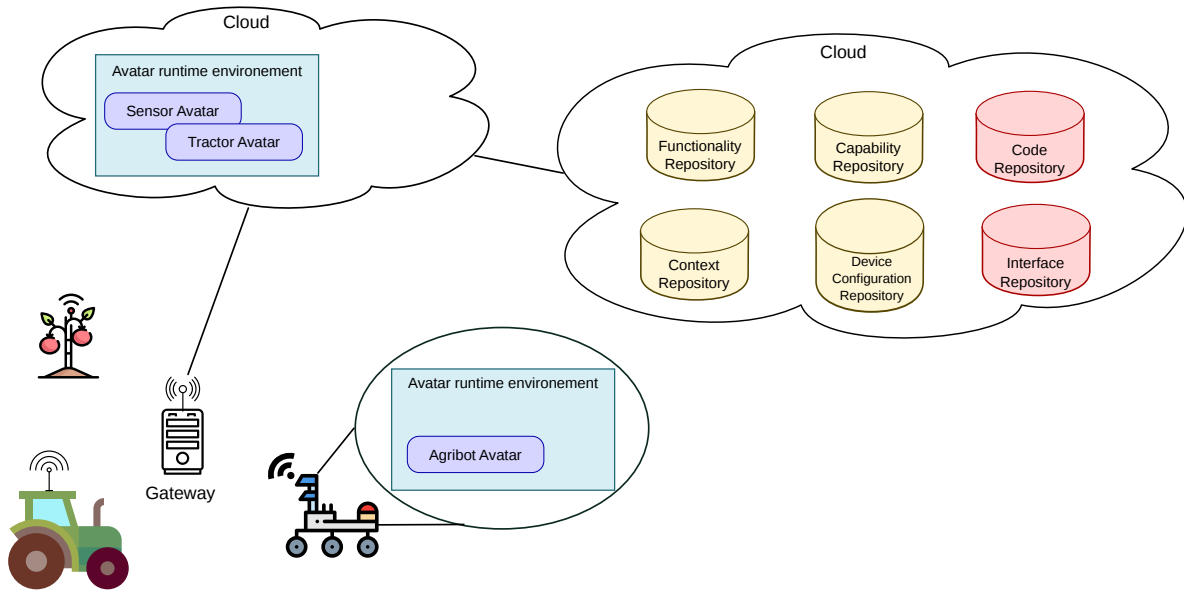


Figure 5. Avatar runtime environment.

320 locally, doing so they instantiate new capabilities and functionalities, and infer new high-level ones progressively.
 321 Avatars are executed in a runtime environment that manages their life-cycle and that exposes them in the Web. An
 322 avatar is instantiated in the runtime environment either automatically when a new physical object is discovered in
 323 the local area network or when a user registers a new device manually. The avatar runtime environment and the
 324 code repositories can be installed either in a cloud platform, on a gateway, or even on a physical object if this
 325 one has enough processing power and memory capacity. Needless to say, the repositories and the avatar runtime
 326 environments can be deployed on separated devices.

327 3.2.3. Technical Implementation

328 The ASAWoO runtime environment is developed in Java and is based on the OSGi Apache Felix platform².
 329 It is designed as a set of services that are deployed using bundles (i.e. a JAR augmented with metadata), and
 330 that manage the life-cycle of avatars. An avatar (i.e. the different modules and managers forming the avatar's
 331 architecture) is also implemented as a set of OSGi services. Configuration files and the pieces of code of
 332 capabilities, functionalities and WoTApps are also deployed using bundles. Code repositories are implemented
 333 by OSGi Bundle Repositories (OBR), which can be either local or remote. An OBR offers a customizable
 334 dependency management that has been tuned to take into account functionalities and capabilities requirements.
 335 The semantic reasoner module depends on Apache Jena³ inference engine and queries Fuseki SPARQL endpoints.
 336 In the current implementation, the Vert.x⁴ stack is used to serve WoTApps and to expose functionalities as REST
 337 services over HTTP. Vert.x handler mechanism also helps in managing asynchronous calls to functionalities. The
 338 opportunistic communication module implementation based on C3PO has already been thoroughly covered in
 339 subsection 3.1.2 and in [31].

340 ASAWoO source code which is distributed under the CeCILL license (compatible with GNU GPLv2), is
 341 available in GIT repository⁵ and users and developers documentation can be accessed on ASAWoO website⁶.

² <http://felix.apache.org/>

³ <https://jena.apache.org/>

⁴ <https://vertx.io/>

⁵ <https://gitlab.com/asawoo>

⁶ <https://asawoo.gitlab.io/>



Figure 6. Montoldre experiment farm.

342 4. Experiment

343 This section describes the experiment conducted during 4 months in Montoldre (France) experimental
344 farm in cooperation with IRSTEA⁷. This experiment aims firstly at evaluating the effective functioning of the
345 ASAWoO middleware platform and its performances in real conditions, and secondly at studying the relevance
346 and the viability of a Web of Thing approach in an agricultural context.

347 4.1. Presentation of the experiment environment

348 Montoldre experimental farm is a large plot of agricultural land dedicated to the experimentation of
349 agricultural techniques, especially fertilizer spreading techniques. The area shown on the satellite view Figure 6
350 is composed of a dozen fields (in red) maintained by a handful of agricultural vehicles.

351 The application considered in this first experiment consisted in collecting environmental data (humidity
352 and temperature) measured by sensors. This experiment was ran in the early stages of ASAWoO development.
353 Even though the whole middleware was not complete at that time, the opportunistic communication module was
354 operational. Hardened sensors that can be exposed to bad weather conditions were also not ready to use for this
355 experiment. Consequently, we used tiny single-board computers (RaspberryPi's) equipped with Wi-Fi antennas
356 that produce data exactly as the sensors would have done. As these devices are not hardened, they were placed in
357 farm buildings, as shown in Figure 7. These experimental conditions may appear as degraded, but those are not
358 in reality because the communication behavior of sensors has been reproduced as accurately as possible. This
359 experiment allowed us to validate the RESTful opportunistic communication support presented in Section 3.

360 Three instances of ASAWoO runtime have been deployed on RaspberryPi's equipped with 5dB Wi-Fi
361 antennas. The first one, identified as RASP00, acted as a base station located in IRSTEA office at the limit of
362 Wi-Fi communication range from the tractors' warehouse. The two other RaspberryPi's (RASP01 and RASP02)
363 were equipped with an external battery pack and a GPS module. The kits were embedded on two tractors.
364 GPS antennas were placed on the tractors' roofs and the battery power cords were plugged into the tractors'
365 cigarette lighters to power up/on the Raspberry and recharge. Two other self-powered RaspberryPi's with smaller
366 range acting as sensors have been deployed in farm buildings located on tractors paths. Locations of the five
367 nodes are shown on the map Figure 7. The embedded system deployed on sensors was restricted to ASAWoO

⁷ <http://www.irstea.fr/en/research/research-units/regions-jru>

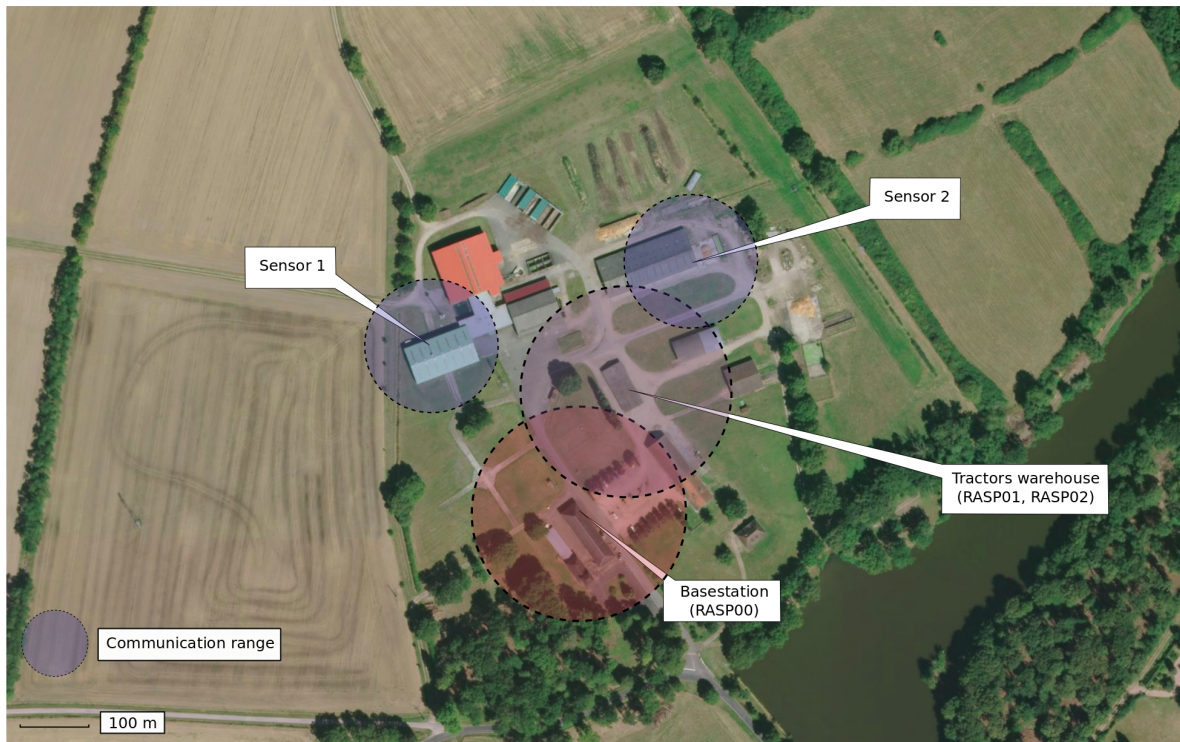


Figure 7. Node locations and their approximate communication ranges.

	Trip duration	Number of messages exchanged			Number of GPS positions	
		RASP00/ RASP01	RASP00/ RASP02	RASP01/ RASP02	RASP01	RASP02
Day 1	5h09	46	110	48	5	147
Day 2	4h50	28	47	3	0	28
Day 3	6h21	56	0	0	266	0
Day 4	7h31	334	0	0	1005	0
Day 5	7h44	219	0	51	0	2063

Table 1. Data of active days.

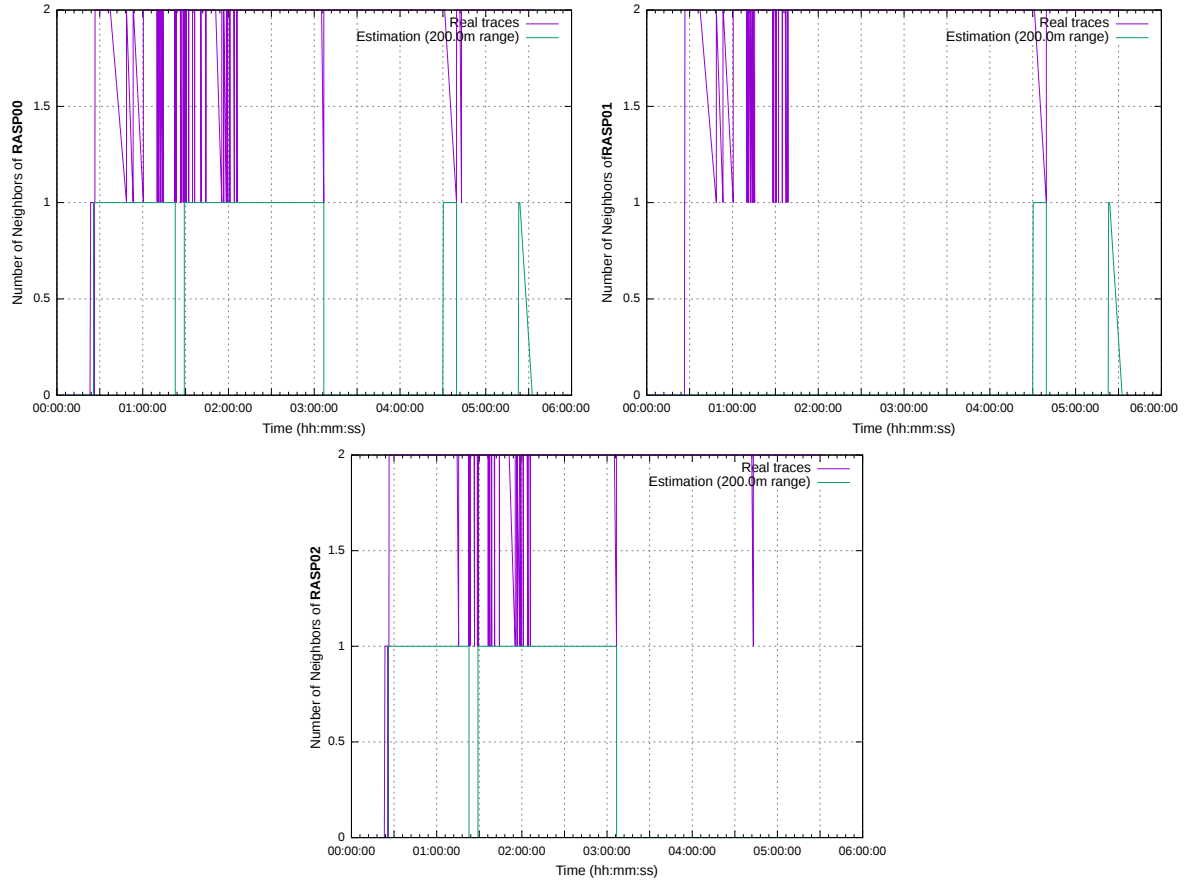
368 communication module since the whole middleware stack was not required. During this experiment, the sensors
 369 produced data every 30 minutes.

370 During the 4 months of experimentation, over 30 trips have been monitored. Mobility traces of RASP01
 371 and RASP02 (i.e. tractors) have been stored in log files, as well as the contacts and exchanges between tractors
 372 and the base station). Logs were compiled and sent to the base station when entering its radio range. A ssh tunnel
 373 between the base station at IRSTEA and our lab has been established in order to access the logs, update the code
 374 if necessary, and remotely monitor the experiment throughout its progress. The following subsection presents the
 375 experiment outcomes, focusing on a few trips involving both tractors or longer activity periods.

376 4.2. Experiment results

377 After 4 months of sparse activity, logs gathered on the base station have been compiled, sanitized and
 378 analyzed. Since the two equipped tractors were rarely operating simultaneously, we focused on the logs of days
 379 providing the most extensive data. A summary of the activity during these days is presented in Table 1. Day
 380 one was the experiment setup, we checked that communication was well established between the three nodes,
 381 and moved RASP01 and RASP02 to make them enter and exit communication range with each other and with
 382 sensors. On day 3, the tractor carrying RASP01 has been spreading fertilizer, and on day 4 it sowed the fields.
 383 Day 5 featured the longer trip from RASP02.

Average speed	Communication range		
	Maximum range	Average range	Median range
15,9 km/h	468 m	237 m	185 m

Table 2. Tractors average speed and radio-contact range.**Figure 8.** Real contacts vs estimated contacts of RASP00, RASP01 and RASP02 on day 1.

384 Table 2 compiles global observations made throughout the experiment. When tractors were active, we
 385 reported an average speed of 16km/h which was not an hindrance to message delivery. Opportunistic sensing and
 386 the data mule approach is therefore a viable solution in the agricultural context where vehicles speed is moderate.
 387 Although cellular network coverage was quite poor on Montoldre site, opportunistic networking via ad hoc Wi-Fi
 388 proved to be effective to convey small messages such as environmental data. Under the right circumstances the
 389 communication range exceeded our expectations with a recorded maximum range of 468 meters. This result is
 390 further supported by the figure 8 which shows the number of actual contacts at a given time between two nodes
 391 versus an estimate derived from the actual node positions and a fixed range of 200 meters. More often than not,
 392 actual contacts were established whereas the estimate did not consider the nodes to be within radio range.

393 Contacts durations (i.e. the time periods during which the nodes are in contact and can communicate) are
 394 presented on Figure 9. On day 5, half of the contacts lasted less than three minutes while only a tenth of the
 395 contacts exceeded 10 minutes. The few contacts exceeding one hour were due to RASP02 instance remaining
 396 active in the warehouse, within range of the base station. The relatively moderate speed of the tractors combined
 397 with the long range Wi-Fi interfaces resulted in long enough contacts allowing for wide communication time
 398 periods. Since the size of exchanged messages, were they gossiping relating to the opportunistic communication
 399 protocol or sensor data, was quite small (a few bytes), such a time frame is sufficient to perform opportunistic
 400 sensing (i.e. collect sensor data and opportunistically forward this data).

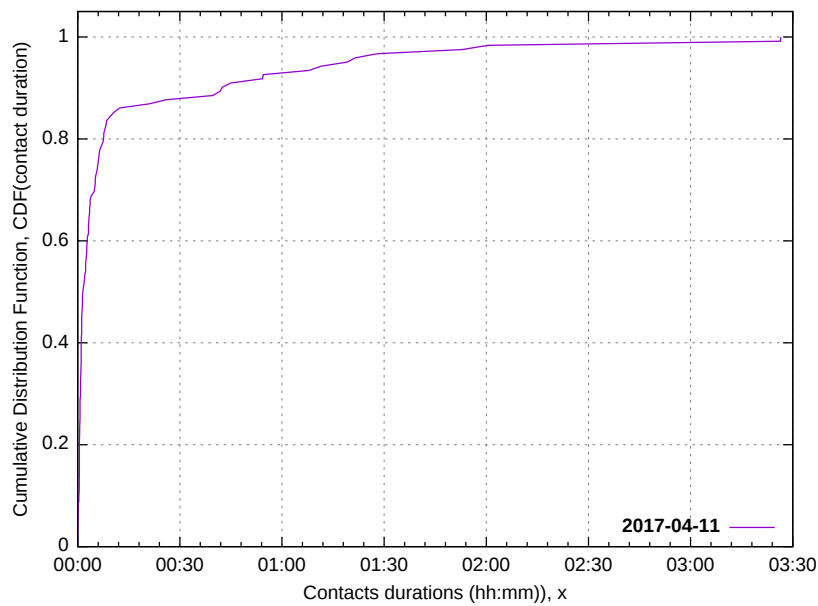


Figure 9. Cumulative distribution of contacts durations on day 5.

401 Nevertheless it is worth to note that the experiment did not go smoothly all the way. Sometimes the
 402 RaspberryPi was slow to receive the GPS signal, which not only led to missing positions, but also to clock
 403 desynchronization and erroneous timestamps since the system clock was set to GPS time. However, logs could be
 404 resynchronized using contacts and exchanged messages with the base station. Due to farming vehicle vibrations,
 405 sometimes the device got its GPS antenna or its power cord unplugged, and the shutdown and powering up that
 406 should have been controlled by the battery was not always done properly, which led to corrupted log files that
 407 had to be discarded or corrected. In a concrete deployment, kits should be designed and built to better fit the
 408 specificities of agricultural vehicles (e.g. with antivibration material, properly integrated power management and
 409 a CMOS clock battery).

410 Lastly, the final encountered issue was the considerable size of the covered area combined with the fact that
 411 the two equipped vehicles were hardly running together, which resulted in too few contacts between the mobile
 412 nodes outside the vicinity of their rallying point (the warehouse). The lack of contact data does not affect the
 413 effectiveness of opportunistic sensing nonetheless. Even though contacts are more likely to occur in the rallying
 414 point area, tractors are still able to retrieve data from faraway sensors and forward it to the base station. The
 415 two mobile ASAWoO instances were indeed able to successfully gather data from sensors up to the base station
 416 without a permanent connectivity between each node.

417 4.3. Evaluation of the Web of Things approach

418 The experiment described in the previous section focused on the validation of ASAWoO opportunistic
 419 communication capabilities whereas the Web of Things aspect could not be tested at that time. WoT modules
 420 were nonetheless still validated in the later stages of ASAWoO development in another experiment which will be
 421 shortly presented in this subsection. During the Montoldre experiment ASAWoO middleware mainly consisted
 422 of the avatar core, the functionalities management and registry features (i.e. functionalities were advertised as
 423 RESTful services) plus the opportunistic communication module. The second experiment was run in Bordeaux,
 424 France. It introduced the device manager and the semantic reasoning features to instantiate capabilities, simple
 425 and composite functionalities, as well as the WoT Application module. Unlike the farm experiment which
 426 spanned over 4 months, this one lasted only one day. Since the communication module had already been validated
 427 in the long-run experiment, the goal was to validate ASAWoO final stage, particularly the WoT related modules
 428 (e.g. WoTApp manager, automatic instantiation of functionalities from device configurations and semantic
 429 repositories, ...).



Figure 10. All-terrain mobile platform running ASAWoO next to the base station kit

430 This experiment was based on the same data collection WoTApp and the same hardware kits. Although
431 the RaspberryPi-based kits (with GPS module and Wi-Fi interface) have been reused, this time the data mule
432 was a robot, the Jackal all-terrain mobile platform⁸, on which was deployed an instance of ASAWoO hosting
433 its own avatar. Figure 10 shows a picture of the robot and the packaged base station kit on the experimentation
434 field. In addition to its navigation capabilities, a camera was mounted on the platform. A ROS⁹ bridge
435 interoperability module was developed for ASAWoO to expose these capabilities, which in turn enabled higher
436 level functionalities such as person detection and video surveillance. The video surveillance and the navigation
437 functionalities were then combined to provide a patrol functionality.

438 The scenario written for this experiment consisted in making the robot patrol from sensor to sensor to
439 collect data as long as it did not encounter a person. If the video surveillance functionality was to trigger an
440 alarm, the reasoning module will give priority to the patrol functionality and use the navigation functionality to
441 go back to a rallying point. This scenario is illustrated on ASAWoO code repository homepage¹⁰.

442 During the experiment, all devices hosting an ASAWoO platform were accessible through their second Wi-Fi
443 interface set on access point mode. When connecting to a device's main WoTApp we could navigate through its
444 neighbors and query their functionalities. Figure 11 presents a screenshot of the main WoTApp hosted on the
445 robot. It features system functionalities (e.g. functionalities registry, neighborhood, device configuration and
446 WoTApp deployment), as well as other avatars whose remote functionalities can be reached through multi-hop
447 asynchronous DTN RESTful requests. The screenshot shows a JSON-formatted GPS position in response to a
448 GET request sent to a sensor's LocationService functionality. The retrieved sensor data were also displayed on
449 the data collection WoTApp.

450 It is worth to note that although this paper mainly focuses on sensing functionalities, ASAWoO also supports
451 actuators and the conclusions that we drew also apply to this class of devices. The general practice that has
452 been followed in ASAWoO was to bind actuator functionalities to PUT and POST HTTP methods while GET
453 methods were used for sensor functionalities. In the patrol surveillance scenario, the navigation functionality is
454 an example of an actuator functionality. Eventually, results regarding context adaptation and semantic reasoning
455 performances were presented in [37].

⁸ Jackal unmanned ground vehicle: <https://www.clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/>

⁹ ROS, Robot Operating System: <http://www.ros.org/>

¹⁰ <https://asawoo.gitlab.io/>

The screenshot displays the ASAWoO administration WoTApp interface. At the top, a blue header contains a menu icon and the text 'API'. Below this, a subtitle reads 'This is the HTTP API exposed by an ASAWoO instance.' The main content area is divided into several sections:

- System::avatar**: A list of endpoints with methods (get, post, delete) and a 'Request' field. A red 'EXECUTE' button is visible. A 'Response' field shows a JSON object: {"longitude": -0.492078565, "latitude": 44.762796963, "altitude": 5.043, ...}. A box labeled 'System fonctionnalités' has arrows pointing to the 'avatar' section and the 'Avatar::sensor2' section.
- Device configurations**: A section with a dropdown arrow.
- Wotapps**: A section with a dropdown arrow.
- System::registry**: A section with a dropdown arrow.
- Avatar::sensor2**: A list of endpoints with methods (get, put, delete). A box labeled 'Remote avatar fonctionnalités' has arrows pointing to the 'sensor2' section and the 'Avatar::husky' section.
- Avatar::husky**: A section with a dropdown arrow.
- Avatar::sensor1**: A section with a dropdown arrow.
- System::wotapp**: A section with a dropdown arrow.
- System::neighborhood**: A section with a dropdown arrow.

Figure 11. Screenshot of ASAWoO administration WoTApp

4.4. Results discussion

During the second experiment, the WoT approach allowed us to successfully interact with different devices through their logical representations. Devices heterogeneity was indeed hidden behind avatars using web standards (JSON over HTTP requests). The web browsers from a laptop, a smartphone or a tablet were used interchangeably.

Both experiments were conclusive regarding the opportunistic computing aspect. The monitored contact durations and distances between mobile and/or static nodes, resulting from the data mules speed and their communication range, both support the idea that opportunistic sensing is a possible option for smart agriculture. The communication range measured in the second experiment confirmed the first results with a maximum range of 283 meters between the base station and the robot. We however observed that farming vehicles were not operating on a daily basis whereas data gathering depends on their operation frequency for an opportunistic sensing approach entirely relies on nodes mobility.

Even though the following perspective is beyond the scope of this article, we argue that the node density issue can be addressed by emulation techniques. At the end of the first experiment, we obtained valuable data on the mobility of farming vehicles in actual conditions. These mobility data could be made available on a service such as CRAWDAD¹¹ and leveraged with simulation tools such as LEPTON [38] which is an emulation platform for opportunistic networking development. LEPTON allows the execution of functional code to run in an emulated environment under controlled and repeatable conditions, where only the mobility of nodes is simulated. The possibility to replay scenarios based on actual tractors paths gives the opportunity to evaluate our approach under different conditions and therefore to finely tune scenario parameters such as the number of data mules, their paths and speed, or the positions of sensors, without starting over an actual experimental campaign.

¹¹ <https://crawdad.org/>

4.77 Thus, testing different smart agriculture scenarios could indeed help in deducing an optimal setup to improve
4.78 opportunistic sensing solutions for precision agriculture.

4.79 5. Conclusion

4.80 The Internet of Things will become a reality when connected object makers will propose off-the-shelves
4.81 cyber-physical objects that can communicate and interoperate through open standard protocols and data formats,
4.82 or that can be connected to Web of Things (WoT) platforms that bridge the gap between proprietary protocols
4.83 and Web standard formats and protocols. Such objects and platforms appear as attractive solutions to build
4.84 complex systems notably for the environmental sensing and for the smart agriculture and farming. Nevertheless
4.85 in challenging environments, such as in agricultural and farming environments, it is often difficult to deploy
4.86 systems relying exclusively on infrastructure-based networks because rural areas have often a poor network
4.87 coverage, and because there are frequent and unpredictable connectivity disruptions resulting from both the
4.88 mobility of agricultural machines (e.g. tractors, agribots, drones) communicating with short range radio interfaces
4.89 and from the sleep phases kept by some devices for energy saving purposes.

4.90 In this paper we presented how opportunistic computing can be combined with the WoT to contribute to
4.91 the smart agriculture, and how such an approach was implemented in a middleware platform called ASAWoO.
4.92 Opportunistic computing and networking techniques leverage the mobility of agricultural machines to provide
4.93 intermittent connectivity between physical objects that would not have been connected by any route otherwise.
4.94 This platform provides a virtual representation of the physical objects in the Web, called avatars, which give
4.95 access to the objects functionalities in order to allow WoT users to interact with devices at an higher level of
4.96 abstraction.

4.97 ASAWoO platform was evaluated in a practical smart agriculture scenario through a 4-month experiment.
4.98 Two tractors carried ASAWoO instances responsible for collecting data of sensors disseminated in an experimental
4.99 farm. Experiment results showed that ASAWoO platform was able to perform environmental sensing using
5.00 opportunistic computing in real conditions, and that the low pace of farming vehicles makes opportunistic
5.01 computing possible in the agricultural context. Based on these outcomes we conclude that both the integration
5.02 of sensors and actuators like agribots in WoT applications, and the implementation of opportunistic computing
5.03 mechanisms are likely to improve agricultural practices and compensate for the potential lack of network
5.04 infrastructure. Such an approach is not limited to smart agriculture and farming, and could also be transposed to
5.05 other related critical contexts such as disaster relief.

5.06 **Author Contributions:** Data curation, Nicolas Le Sommer; Investigation, Lionel Touseau; Resources, Nicolas Le Sommer;
5.07 Software, Lionel Touseau and Nicolas Le Sommer; Supervision, Nicolas Le Sommer; Validation, Lionel Touseau;
5.08 Visualization, Lionel Touseau; Writing - original draft, Lionel Touseau and Nicolas Le Sommer; Writing - review &
5.09 editing, Lionel Touseau and Nicolas Le Sommer.

5.10 **Funding:** This work was supported by the French ANR (Agence Nationale de la Recherche) grant number
5.11 ANR-13-INFR-012.

5.12 **Acknowledgments:** The experiment presented in this article would not have been possible without the helpful support
5.13 provided by IRSTEA from Montoldre and Clermont-Ferrand. We would also like to thank other partners for their valuable
5.14 contributions to the ASAWoO project: TWEAK team from Lyon 1 university LIRIS lab (<http://liris.cnrs.fr/>), COSY team
5.15 from Valence-Grenoble LCIS lab (<http://lcis.grenoble-inp.fr/>), and Generation Robots (<https://generationrobots.com/>).

5.16 References

- 5.17 1. Li, Z.; Wang, J.; Higgs, R.; Zhou, L.; Yuan, W. Design of an Intelligent Management System for Agricultural
5.18 Greenhouses Based on the Internet of Things. *2017 IEEE International Conference on Computational Science and
5.19 Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC) 2017*, pp.
5.20 154–160.
- 5.21 2. Vasisht, D.; Kapetanovic, Z.; Won, J.; Jin, X.; Chandra, R.; Sinha, S.; Kapoor, A.; Sudarshan, M.; Stratman, S.
5.22 FarmBeats: An IoT Platform for Data-Driven Agriculture. *14th USENIX Symposium on Networked Systems Design
5.23 and Implementation (NSDI 17)*; USENIX Association: Boston, MA, 2017; pp. 515–529.
- 5.24 3. Sreekantha, D.K.; Kavya, A.M. Agricultural crop monitoring using IOT - A study. *Proceedings of 2017 11th
5.25 International Conference on Intelligent Systems and Control, ISCO 2017 2017*, pp. 134–139.

- 526 4. Zhao, D.; Ma, H.; Liu, L.; Zhao, J. On Opportunistic Coverage for Urban Sensing. Proceedings of 10th International
527 Conference on Mobile Adhoc and Sensor Systems (MASS 2013); IEEE: Hangzhou, China, 2013; pp. 218–226.
- 528 5. Guinard, D.; Trifa, V.; Mattern, F.; Wilde, E. From the Internet of Things to the Web of Things: Resource-oriented
529 Architecture and Best Practices. In *Architecting the Internet of Things*; Uckelmann, D.; Harrison, M.; Michahelles, F.,
530 Eds.; Springer, 2011; pp. 97–129. doi:10.1007/978-3-642-19157-2.
- 531 6. Guinard, D.; Trifa, V.; Wilde, E. A Resource Oriented Architecture for the Web of Things. IEEE International
532 Conference on the Internet of Things (IOT 2010), 2010, pp. 1–8. doi:10.1109/IOT.2010.5678452.
- 533 7. Wilde, E. Putting Things to REST. Technical report, School Of Information and UC Berkeley, 2007.
- 534 8. Mrissa, M.; Médini, L.; Jamont, J.P.; Le Sommer, N.; Laplace, J. An Avatar Architecture for the Web of Things.
535 *Internet Computing, IEEE* **2015**, pp. 30–38.
- 536 9. Rehman, A.u.; Abbasi, A.; Islam, N.; Shaikh, Z. A Review of Wireless Sensors and Networks' Applications in
537 Agriculture **2014**. *36*, 263–270.
- 538 10. Ojha, T.; Misra, S.; Raghuvanshi, N.S. Wireless sensor networks for agriculture: The state-of-the-art in practice and
539 future challenges. *Computers and Electronics in Agriculture* **2015**, *118*, 66–84.
- 540 11. Kim, Y.; Evans, R.G.; Iversen, W.M. Remote Sensing and Control of an Irrigation System Using a Distributed
541 Wireless Sensor Network. *IEEE Transactions on Instrumentation and Measurement* **2008**, *57*, 1379–1387.
- 542 12. Hull, B.; Bychkovsky, V.; Zhang, Y.; Chen, K.; Goraczko, M.; Miu, A.; Shih, E.; Balakrishnan, H.; Madden, S.
543 CarTel: A Distributed Mobile Sensor Computing System. Proceedings of the 4th International Conference on
544 Embedded Networked Sensor Systems; ACM: Boulder, Colorado, USA, 2006; SenSys '06, pp. 125–138.
- 545 13. Campbell, A.T.; Eisenman, S.B.; Lane, N.D.; Miluzzo, E.; Peterson, R.A. People-centric Urban Sensing. Proceedings
546 of the Second Annual International Wireless Internet Conference; ACM Press: Boston, Massachusetts, USA, 2006;
547 WICON '06, pp. 18–31.
- 548 14. Abdelzaher, T.; Anokwa, Y.; Boda, P.; Estrin, D.; Burke, J.; Leonidas, G.; Madden, S.; Reich, J. Mobiscopes for
549 Human Spaces. *IEEE Pervasive Computing* **2007**, *6*, 20–29.
- 550 15. Kansal, A.; Nath, S.; Liu, J.; Zhao, F. SenseWeb: An Infrastructure for Shared Sensing. *IEEE MultiMedia* **2007**,
551 *14*, 8–13.
- 552 16. Riva, O.; Borcea, C. The Urbanet Revolution: Sensor Power to the People! *IEEE Pervasive Computing* **2007**,
553 *6*, 41–49.
- 554 17. Urban Atmospheres. <http://www.urban-atmospheres.net/>.
- 555 18. Eisenman, S.B.; Miluzzo, E.; Lane, N.D.; Peterson, R.A.; Ahn, G.S.; Campbell, A.T. The BikeNet Mobile Sensing
556 System for Cyclist Experience Mapping. Proceedings of the 5th International Conference on Embedded Networked
557 Sensor Systems; ACM: Sydney, Australia, 2007; SenSys '07, pp. 87–101.
- 558 19. Miluzzo, E.; Lane, N.D.; Eisenman, S.B.; Campbell, A.T. CenceMe - Injecting Sensing Presence into Social
559 Networking Applications. Smart Sensing and Context, Second European Conference; Springer: Kendal, England,
560 2007; EuroSSC 2007, pp. 1–28.
- 561 20. MIT Senseable City Lab.
- 562 21. Chen, K.W. CafNet: A Carry-and-Forward Delay-Tolerant Network. Master's thesis, Massachusetts Institute of
563 Technology, 2007.
- 564 22. Mota, V.F.S.; Cunha, F.D.; Macedo, D.F.; Nogueira, J.M.S.; Loureiro, A.A.F. Protocols, Mobility Models and Tools
565 in Opportunistic Networks: A Survey. *Computer Communications* **2014**, *48*, 5–19.
- 566 23. Kamilaris, A.; Gao, F.; Prenafeta-Boldu, F.X.; Ali, M.I. Agri-IoT: A semantic framework for Internet of
567 Things-enabled smart farming applications. 3rd IEEE World Forum on Internet of Things (WF-IoT 2016); ,
568 2016; pp. 442–447. doi:10.1109/WF-IoT.2016.7845467.
- 569 24. Taylor, K.; Griffith, C.; Lefort, L.; Gaire, R.; Compton, M.; Wark, T.; Lamb, D.; Falzon, G.; Trotter, M. Farming the
570 Web of Things. *IEEE Intelligent Systems* **2013**, *28*, 12–19. doi:10.1109/MIS.2013.102.
- 571 25. Kamilaris, A.; Trifa, V.; Pitsillides, A. HomeWeb: An application framework for Web-based smart homes. 18th
572 International Conference on Telecommunications (ICT 2011); , 2011; pp. 134–139. doi:10.1109/CTS.2011.5898905.
- 573 26. Kamilaris, A.; Pitsillides, A.; Trifa, V. The Smart Home meets the Web of Things. *International Journal of Ad Hoc
574 and Ubiquitous Computing (IJAHUC)* **2011**, *7*, 145–154. doi:10.1504/IJAHUC.2011.040115.
- 575 27. Fall, K. A Delay-Tolerant Network Architecture for Challenged Internets. Proceedings of ACM SIGCOMM03; ,
576 2003; pp. 27–34.
- 577 28. Pelusi, L.; Passarella, A.; Conti, M. Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc
578 Networks. *IEEE Communications Magazine* **2006**, *44*, 134–141.

- 579 29. Conti, M.; Giordano, S.; May, M.; Passarella, A. From Opportunistic Networks to Opportunistic Computing. *IEEE*
580 *Communications Magazine* **2010**, *48*, 126–139.
- 581 30. Juang, P.; Oki, H.; Wang, Y.; Martonosi, M.; Peh, L.S.; Rubenstein, D. Energy-efficient Computing for Wildlife
582 Tracking: Design Tradeoffs and Early Experiences with ZebraNet. Proceedings of the 10th International Conference
583 on Architectural Support for Programming Languages and Operating Systems (APLOS X); ACM: San Jose, CA,
584 USA, 2002; pp. 96–107.
- 585 31. Le Sommer, N.; Touseau, L.; Mahéo, Y.; Auzias, M.; Raimbault, F. A Disruption-Tolerant RESTful Support for the
586 Web of Things. 4th International Conference on Future Internet of Things and Cloud (FiCloud 2016); Younas, M.;
587 Awan, I.; Seah, W., Eds.; , 2016; pp. 17 – 24.
- 588 32. Scott, K.; Burleigh, S. Bundle Protocol Specification. IETF RFC 5050, 2007.
- 589 33. Cerf, V.G.; Burleigh, S.C.; Durst, R.C.; Fall, K.; Hooke, A.J.; Scott, K.L.; Torgerson, L.; Weiss, H.S. Delay-Tolerant
590 Networking Architecture. IETF RFC 4838, 2007.
- 591 34. Le Sommer, N.; Launay, P.; Mahéo, Y. A Framework for Opportunistic Networking in Spontaneous and Ephemeral
592 Social Networks. 10th Workshop on Challenged Networks (CHANTS'2015); ACM: Paris, France, 2015.
- 593 35. Médini, L.; Mrissa, M.; Terdjimi, M.; Khalfi, E.M.; Le Sommer, N.; Capdepuy, P.; Jamont, J.P.; Ocello, M.; Touseau,
594 L. Building a Web of Things with Avatars. In *Managing the Web of Things: Linking the Real World to the Web*;
595 Shengn, M.; Qin, Y.; Yao, L.; Benatallah, B., Eds.; Elsevier, 2016.
- 596 36. W3C. Web of Things (WoT) Architecture, 2017.
- 597 37. Terdjimi, M.; Médini, L.; Mrissa, M.; Le Sommer, N. An Avatar-Based Adaptation Workflow for the Web of Things.
598 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises
599 (WETICE); , 2016. doi:10.1109/WETICE.2016.22.
- 600 38. Sánchez-Carmona, A.; Guidec, F.; Launay, P.; Mahéo, Y.; Robles, S. Filling in the missing link between
601 simulation and application in opportunistic networking. *Journal of Systems and Software* **2018**, *142*, 57 – 72.
602 doi:10.1016/j.jss.2018.04.025.